ZILKA·KOTAB
PC
ZILKA, KOTAB & FEECE™

95 SOUTH MARKET ST., SUITE 420
SAN JOSE, CA 95113

TELEPHONE (408) 971-2573
FAX (408) 971-4660

FAX COVER SHEET

		Phone Number	Fax Number
Date:	September 29, 2004		
To:	Appeal Briefs- Patents		(703) 305-0942
From:	Kevin J. Zilka		

Docket No.: NAI1P194_99.115.01

App. No: 09/664,919

Total Number of Pages Being Transmitted, Including Cover Sheet: 33

Message:

Please deliver to the Board of Patent Appeals and Interferences.

Thank you,

Kevin J. Zilka

□ *Original to follow Via Regular Mail* X *Original will Not be Sent* □ *Original will follow Via Overnight Courier*

****

The information contained in this facsimile message is attorney privileged and confidential information intended only for the use of the individual or entity named above. If the reader of this message is not the intended recipient, you are hereby notified that any dissemination, distribution or copy of this communication is strictly prohibited. If you have received this communication in error, please immediately notify us by telephone (if long distance, please call collect) and return the original message to us at the above address via the U.S. Postal Service. Thank you.

****

IF YOU DO NOT RECEIVE ALL PAGES OR IF YOU ENCOUNTER
ANY OTHER DIFFICULTY, PLEASE PHONE Erica
AT (408) 971-2573 AT YOUR EARLIEST CONVENIENCE

Practitioner's Docket No. NAI1P194/99.115.01 ***PATENT***

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Drew

Application No.: 09/664,919 Group No.: 2131
Filed: 9/18/2000 Examiner: Colin, C.
For: METHOD AND APPARATUS FOR MINIMIZING FILE SCANNING BY ANTI-VIRUS PROGRAMS

Mail Stop Appeal Briefs – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

1. Transmitted herewith is the APPEAL BRIEF in this application, with respect to the Notice of Appeal filed on September 27, 2004.

2. STATUS OF APPLICANT

This application is on behalf of other than a small entity.

CERTIFICATION UNDER 37 C.F.R. §§ 1.8(a) and 1.10*

*(When using Express Mail, the Express Mail label number is **mandatory**; Express Mail certification is optional.)*

I hereby certify that, on the date shown below, this correspondence is being:

MAILING

_ deposited with the United States Postal Service in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

37 C.F.R. § 1.8(a)
_ with sufficient postage as first class mail.

37 C.F.R. § 1.10*
_ as "Express Mail Post Office to Addressee"
Mailing Label No. ____________________ **(mandatory)**

TRANSMISSION

✓ facsimile transmitted to the Patent and Trademark Office, (703) 305 - 0942.

Signature

Date: 9|29|2004

Erica L. Farlow
(type or print name of person certifying)

*** Only the date of filing (' 1.6) will be the date used in a patent term adjustment calculation, although the date on any certificate of mailing or transmission under ' 1.8 continues to be taken into account in determining timeliness. See ' 1.703(f). Consider "Express Mail Post Office to Addressee" (' 1.10) or facsimile transmission (' 1.6(d)) for the reply to be accorded the earliest possible filing date for patent term adjustment calculations.***

3. FEE FOR FILING APPEAL BRIEF

Pursuant to 37 C.F.R. § 1.17(c), thc fee for filing the Appeal Brief is:

other than a small entity	$330.00

Appeal Brief fee due $330.00

4. EXTENSION OF TERM

The proceedings herein are for a patent application and the provisions of 37 C.F.R. § 1.136 apply.

Applicant believes that no extension of term is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

5. TOTAL FEE DUE

The total fee due is:

Appeal brief fee	$330.00
Extension fee (if any)	$0.00

TOTAL FEE DUE $330.00

6. PAYMENT OF FEES

The commissioner is authorized to charge deposit account 50-1351 (NAI1P194) in the amount of $330.00. A duplicate of this transmittal is attached.

7. FEE DEFICIENCY

If any additional extension and/or fee is required, and if any additional fee for claims is required, charge Deposit Account No. 50-1351 (Order No. NAI1P194).

Signature of Practitioner

Reg. No.: 41,429
Tel. No.: 408-971-2573
Customer No.: 28875

Kevin J. Zilka
Silicon Valley IP Group, PC
P.O. Box 721120
San Jose, CA 95172-1120
USA

Practitioner's Docket No. NAI1P194/99.115.01 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Drew

Application No.: 09/664,919 Group No.: 2131
Filed: 9/18/2000 Examiner: Colin, C.
For: METHOD AND APPARATUS FOR MINIMIZING FILE SCANNING BY ANTI-VIRUS PROGRAMS

Mail Stop Appeal Briefs – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

1. Transmitted herewith is the APPEAL BRIEF in this application, with respect to the Notice of Appeal filed on September 27, 2004.

2. STATUS OF APPLICANT

   This application is on behalf of other than a small entity.

CERTIFICATION UNDER 37 C.F.R. §§ 1.8(a) and 1.10*

(When using Express Mail, the Express Mail label number is ***mandatory;*** *Express Mail certification is optional.)*

I hereby certify that, on the date shown below, this correspondence is being:

MAILING

_ deposited with the United States Postal Service in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

37 C.F.R. § 1.8(a)
_ with sufficient postage as first class mail.

37 C.F.R. § 1.10*
_ as "Express Mail Post Office to Addressee"
Mailing Label No. ____________________ **(mandatory)**

TRANSMISSION

✓facsimile transmitted to the Patent and Trademark Office, (703) 305 - 0942.

Erica L. Farlow
Signature

Date: 9/29/2004

Erica L. Farlow
(type or print name of person certifying)

** Only the date of filing (' 1.6) will be the date used in a patent term adjustment calculation, although the date on any certificate of mailing or transmission under ' 1.8 continues to be taken into account in determining timeliness. See ' 1.703(f). Consider "Express Mail Post Office to Addressee" (' 1.10) or facsimile transmission (' 1.6(d)) for the reply to be accorded the earliest possible filing date for patent term adjustment calculations.*

3. FEE FOR FILING APPEAL BRIEF

Pursuant to 37 C.F.R. § 1.17(c), the fee for filing the Appeal Brief is:

other than a small entity	$330.00

Appeal Brief fee due $330.00

4. EXTENSION OF TERM

The proceedings herein are for a patent application and the provisions of 37 C.F.R. § 1.136 apply.

Applicant believes that no extension of term is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

5. TOTAL FEE DUE

The total fee due is:

Appeal brief fee	$330.00
Extension fee (if any)	$0.00

TOTAL FEE DUE $330.00

6. PAYMENT OF FEES

The commissioner is authorized to charge deposit account 50-1351 (NAI1P194) in the amount of $330.00. A duplicate of this transmittal is attached.

7. FEE DEFICIENCY

If any additional extension and/or fee is required, and if any additional fee for claims is required, charge Deposit Account No. 50-1351 (Order No. NAI1P194).

Signature of Practitioner

Reg. No.: 41,429
Tel. No.: 408-971-2573
Customer No.: 28875

Kevin J. Zilka
Silicon Valley IP Group, PC
P.O. Box 721120
San Jose, CA 95172-1120
USA

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:)
)
Drew) Art Unit: 2131
)
Application No. 09/664,919) Examiner: Colin, C.
)
Filed: 09/18/2000) September 29, 2004
)
For:)
METHOD AND APPARATUS FOR MINIMIZING)
FILE SCANNING BY ANTI-VIRUS PROGRAMS)

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

APPELLANT'S BRIEF (37 C.F.R. § 41.37)

This brief is in furtherance of thc Notice of Appeal, filed in this case on September 27, 2004.

The fees required under § 1.17, and any required petition for extension of time for filing this brief and fees therefor, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief contains these items under the following headings, and in the order set forth below (37 C.F.R. § 41.37(c)(i)):

I REAL PARTY IN INTEREST
II RELATED APPEALS AND INTERFERENCES
III STATUS OF CLAIMS
IV STATUS OF AMENDMENTS

V SUMMARY OF CLAIMED SUBJECT MATTER

VI ISSUES

VII ARGUMENTS

VIII APPENDIX OF CLAIMS INVOLVED IN THE APPEAL

IX APPENDIX LISTING ANY EVIDENCE RELIED ON BY THE APPELLANT IN THE APPEAL

The final page of this brief bears the practitioner's signature.

I REAL PARTY IN INTEREST (37 C.F.R. § 41.37(c)(1)(i))

The real party in interest in this appeal is Networks Associates Technology, Inc.

II RELATED APPEALS AND INTERFERENCES (37 C.F.R. § 41.37(c) (1)(ii))

With respect to other prior or pending appeals, interferences, or related judicial proceedings that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no other such appeals, interferences, or related judicial proceedings.

Since no such proceedings exist, no Related Proceedings Appendix is appended hereto.

III STATUS OF CLAIMS (37 C.F.R. § 41.37(c) (1)(iii))

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-15.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims withdrawn from consideration but not canceled: None
2. Claims pending: 1-15
3. Claims allowed: None
4. Claims rejected: 1-15

C. CLAIMS ON APPEAL

The claims on appeal are: 1-15

See additional status information in the Appendix of Claims.

IV STATUS OF AMENDMENTS (37 C.F.R. § 41.37(c)(1)(iv))

As to the status of any amendment filed subsequent to final rejection, there are no such amendments after final.

V SUMMARY OF CLAIMED SUBJECT MATTER (37 C.F.R. § 41.37(c)(1)(v))

With respect to a summary of Claim 1 et al., a method and computer program product are provided for optimizing the operation of an anti-virus computer program for use with an operating system. Initially, a request for closure of an opened computer file is detected. As shown in operation 40 of Figures 3 and 4, for example, it is then determined in response to and after a closure request, but before file closure, if the opened computer file has been modified since being opened. If the opened file has been modified, the opened file is scanned for viruses before closure. See operation 30 of Figures 3 and 4, for example. Thereafter, the file is closed if unmodified, or after scanning for viruses (if found virus free). Note, for example, page 7, line 20 – page 9, line 25.

With respect to a summary of Claims 7-8 et al., the above summary is incorporated, at least in part, by reference. Further provided is an operating system including programming for raising a flag indicative of modification of an open file during the time the file has been open. See, for example, operation 40 of Figures 3 and 4, and page 8, lines 1-15.

With respect to a summary of Claims 9 and 12 et al., the above summary is incorporated, at least in part, by reference. Further included is computer code for, after the detecting operation, but before file closure, accessing an operating system flag that indicates whether the requested file was changed prior to the close request. See, for example, operation 40 of Figures 3 and 4, and page 8, lines 1-15. Still yet, further provided is computer code for scanning the requested file for computer viruses if the requested file was changed prior to the close request; and computer code for skipping scanning the requested file if it was not changed prior to the close request. See, for example, operations 30 and 42 of Figures 3 and 4, and page 9, line 10 – page 11, line 2.

With respect to a summary of Claim 15 et al., the above summary is incorporated, at least in part, by reference. Still yet, a cache buffer memory is established for storing upon opening of a file only a virus vulnerable portion of that file that a virus must use to enter and infect the file. Such modification determining step further includes the steps of: indicating an open file is unmodified in the absence of an associated modification flag; responding to the presence of a modification flag by comparing a portion of the open file to the associated unmodified virus vulnerable portion of the file in the cache buffer memory to determine if the portion of the open file has been modified since the opening of the file; indicating the opened file is unmodified if the virus vulnerable portion is unmodified; and indicating the opened file is modified if the virus vulnerable portion is modified. See, for example, page 8, lines 17-25, and page 10, line 7 – page 11, line 8.

VI ISSUES (37 C.F.R. § 41.37(c)(1)(vi))

Following, under each issue listed, is a concise statement setting forth the corresponding ground of rejection.

Issue # 1: The Examiner has rejected Claims 1-4, and 6-8 under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,398,196 to Chambers in view of International Application Publication WO-93/25024 (PCT/US93/05029) to Branham and further in view of appellant's admitted prior art (AAPA).

Issue # 2: The Examiner has rejected Claims 9-14 under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,319,776 to Hile et al. in view of International Application Publication WO-93/25024 (PCT/US93/05029) to Branham and further in view of appellant's admitted prior art (AAPA).

Issue # 3: The Examiner has rejected Claims 5 and 15 under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,398,196 to Chambers in view of International Application Publication WO-93/25024 (PCT/US93/05029) to Branham in view of appellant's admitted prior art (AAPA) and further in view of US Patent 5,649,095 to Cozza.

VII ARGUMENTS (37 C.F.R. § 41.37(c)(1)(vii))

The claims of the groups noted below do not stand or fall together. In the present section, appellant explains why the claims of each group are believed to be separately patentable.

Issue #1:

The Examiner has rejected Claims 1-4, and 6-8 under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,398,196 to Chambers in view of International Application Publication WO-93/25024 (PCT/US93/05029) to Branham and further in view of appellant's admitted prior art (AAPA).

Group #1: Claim 1

With respect to the present group, the Examiner relies on page 6, line 16 through page 7, line 15; page 8, line 4 through page 8, line 27; page 3, lines 15-26; and page 5, lines 20-32 of Branham to make a prior art showing of appellant's claimed "determining in response to and after a closure request, but before file closure, if the opened computer file has been modified since being opened."

Branham, however, merely suggests various security techniques that are initiated as a function of a file being modified. There is simply no disclosure, teaching or suggestion of making a determination as to whether a file is modified relative to the open and closure thereof, as claimed. Specifically, Branham is completely devoid of any sort of "determining in response to and after a **closure request**, but before **file closure**, if the **opened** computer file has been modified **since being opened**" (emphasis added).

Only appellant teaches and claims such feature which enables the presently claimed invention to minimize the scanning of an opened filed for viruses between a time a user requests closure of the file, and the time the file is actually closed.

It appears that the Examiner continues to rely on appellant's admitted prior art (AAPA) to make a prior art showing of appellant's claimed "scanning said opened file for viruses before closure only if said opened file has been modified." Specifically, the Examiner argues that AAPA "discloses that an opened file is scanned before closure."

In response, appellant notes that, even if the Examiner's assertion was correct, AAPA still does not provide for a technique for "scanning said opened file for viruses before closure <u>only if said opened file has been modified</u>" (emphasis added). Again, such feature is paramount for minimizing the scanning of an opened filed for viruses between a time a user requests closure of the file, and the time the file is actually closed.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on appellant's disclosure. *In re Vaeck,*947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

Appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has simply not been met, <u>since the prior art references, when combined, fail to teach or suggest all the claim limitations</u>.

Group #2: Claims 2 and 3

With respect to the above group, it is further noted that the Examiner's application of the prior art to appellant's dependent claims is replete with deficiencies. Just by way of example, the Examiner has rejected Claims 2-3 by relying on col. 3, line 64 – col. 4, line 3; col. 10, lines 7-14; and col. 9, 11-60 of Chambers.

After careful review of such excerpts, however, appellant asserts that there is simply no suggestion of:

> "before said detecting step, the steps of:
>
> determining whether said operating system includes a "dirty cache buffer" to raise or set a modification flag relative to a file being modified during the time it has been open, a computer code being indicative of said flag; and
>
> using the computer code for a raised or set modification flag, if available, for carrying out said modification determining step by checking for the presence of a raised modification for said file" (see Claim 2 – emphasis added); and
>
> "wherein if it is determined that said operating system does not provide a file modification flag, said method further includes the steps of:
>
> establishing a "dirty cache buffer"; and
>
> raising a modification flag in said "dirty cache buffer" if an opened file associated with said flag has been modified by a write operation" (see Claim 3 – emphasis added).

Appellant respectfully asserts that the mere isolated disclosure of "temporary storage" and a "flag" in no way meets the specific functionality claimed. Specifically, Chambers merely discloses a flag that is "set to indicate that the entry point has changed, so that the change will not be logged redundantly later." In stark contrast, appellant teaches and claims, for example, a modification flag that is set or raised "relative to a file being modified **during the time it has been open**, a computer code being indicative of said flag; and using the computer code for a raised or set modification flag, if available, for carrying out said modification determining step by checking for the presence of a raised modification for said file" (emphasis added). Only appellant teaches and claims such a flag that indicates modification of a file in a specific state, namely during the time such file has been open, in the context of the remaining claim limitations.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has simply not been met, since the prior art references, when combined, fail to teach or suggest all the claim limitations.

Group #3: Claim 4

With respect to Claim 4, the Examiner relies on col. 10, lines 7-14; and col. 9, lines 11-60 of Chambers to make a prior art showing of such limitations. After careful review of such excerpts, however, appellant asserts that there is simply no suggestion of:

> "wherein said operating system includes a "dirty cache buffer" for providing a computer code for a modification flag indicative of the modification **of an open file**, said method further including in said modification determining step, the step of:
>
> detecting the presence of said modification flag to determine if the associated opened file has been modified" (see Claim 4 – emphasis added.

Again, Appellant respectfully asserts that the mere isolated disclosure of "temporary storage" and a "flag" in no way meets the specific functionality claimed. Specifically, Chambers merely discloses a flag that is "set to indicate that the entry point has changed, so that the change will not be logged redundantly later." In stark contrast, appellant teaches and claims, for example, "a modification flag indicative of the modification of an open file, said method further including in said modification determining step, the step of: detecting the presence of said modification flag to determine if the associated opened file has been modified" (emphasis added). Only appellant teaches and claims such a flag that indicates modification of a file in a specific state, namely during the time such file has been open, in the context of the remaining claim limitations.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has simply not been met, since the prior art references, when combined, fail to teach or suggest all the claim limitations.

Group #4: Claim 6

With respect to Claim 6, it appears that the Examiner has not made a specific prior art showing of appellant's claimed "wherein said step of determining in response to a closing request if the opened computer file has been modified since being opened includes the step of: monitoring network protocols to determining if a write packet was initiated for a given open file." After careful review of Chambers, Branham, and AAPA, appellant can not find such claimed features in the prior art.

Appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has simply not been met, since the prior art references, when combined, fail to teach or suggest all the claim limitations.

Group #5: Claims 7-8

With respect to such group, the Examiner relies on the following excerpts of Chambers to make a prior art showing of appellant's claimed "said operating system including programming for raising a flag indicative of modification of an open file during the time the file has been open ..."

> "As shown in FIG. 9, when the interrupt handler routine is entered at block 900, the first action is to open the guinea pig file, at block 910, after which the guinea pig file is closed at block 920. Next, at block 930 the interrupt handler testing routine examines the guinea pig

> file to determine if its content has been changed. Such would be the result of a virus having contaminated the interrupt handlers for opening or closing files." (col. 10, lines 7-14)
>
> "Controlling Access to Operating System Entry Points
>
> Block 550 is illustrated in further detail by FIG. 8. This control of operating system entry points need not be performed to obtain substantial benefits from the emulation of the target program; however, this process does a higher level of control over the target program and also allows for a more accurate evaluation of viral behavior on the part of the target program.
>
> From beginning block 800 control passes to block 810, at which the monitor program examines a list of operating system entry points to determine if any have changed as a result of the instruction just emulated. This would indicate that the target program had replaced an interrupt handler with a routine of its own. If there is such a change, then it is logged at block 820. At block 820 a flag is also preferable set to indicate that the entry point has changed, so that the change will not be logged redundantly later. In some embodiments, the flag indicates the new value of the entry point, so the monitor program can determine if the entry point gets modified yet again.
>
> After block 820, at block 830 the emulated instruction pointer, emulated code segment register, and emulated flag register are saved onto the emulated stack. Then the emulated stack pointer is decremented the corresponding 6 bytes, in the same manner as if a hardware interrupt had been received. Next, at block 840, the emulated code segment register and emulated instruction pointer are set to a special purpose monitor program routine to test the interrupt handler just installed by the target program. This interrupt handler testing routine is described below with reference to FIG. 9.
>
> After block 840, execution passes to block 850, which returns control to the basic process of FIG. 5. This causes the interrupt handler routine of FIG. 9 to be emulated in the same step by step manner as the target program. This maintains the highest degree of encapsulation around the target program, although if detecting viral replication is essentially the only concern, the interrupt handler testing routine of FIG. 9 may alternatively be executed in a more straightforward emulation without many of the execution safeguards described above.
>
> If at block 810 the monitor program had determined that no operating system entry points had been changed, then control would have passed directly to block 850, and thus returned to the process of FIG. 5 to emulate the remainder of the target program." (col. 9, lines 11-60)

Yet again, appellant respectfully asserts that the mere isolated disclosure of "temporary storage" and a "flag" in no way meets the specific functionality claimed. Specifically, Chambers merely discloses a flag that is "set to indicate that the entry point has changed, so that the change will not be logged redundantly later." In stark contrast, appellant teaches and claims, for example "said operating system including programming for raising a flag indicative of modification <u>of an open file during the time the file has been open</u> ..." (emphasis added). Only appellant teaches and claims such a flag that indicates modification of a file in a specific state, namely during the time such file has been open, in the context of the remaining claim limitations.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has simply not been met, since the prior art references, when combined, fail to teach or suggest all the claim limitations.

Issue # 2:

The Examiner has rejected Claims 9-14 under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,319,776 to Hile et al. in view of International Application Publication WO-93/25024 (PCT/US93/05029) to Branham and further in view of appellant's admitted prior art (AAPA).

Group #1: Claims 9-14

With respect to the present group, the Examiner relies on page 6, line 16 through page 7, line 15; page 8, line 4 through page 8, line 27; page 3, lines 15-26; and page 5, lines 20-32 of Branham to make a prior art showing of appellant's claimed "computer code for, after said detecting, but before file closure, accessing operating system flag that indicates whether the requested file was changed prior to said close request; computer code for scanning said requested file for computer viruses if said requested file was changed prior to said close request; and computer code for skipping scanning said requested file if it was not changed prior to said close request" (see this and similar language in Claims 9 and 12).

Branham, however, merely suggests various security techniques that are initiated as a function of a file being modified. There is simply no disclosure, teaching or suggestion of making a determination as to whether a file is modified relative to the open and closure thereof, as claimed. Specifically, Branham is completely devoid of any sort of "after said detecting, but before file **closure**, accessing operating system flag that indicates whether the requested file was changed **prior to said close request**;" (emphasis added).

Only appellant teaches and claims such feature which enables the presently claimed invention to minimize the scanning of an opened filed for viruses between a time a user requests closure of the file, and the time the file is actually closed.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has simply not been met, since the prior art references, when combined, fail to teach or suggest all the claim limitations.

Issue # 3:

The Examiner has rejected Claims 5 and 15 under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,398,196 to Chambers in view of International Application Publication WO-93/25024 (PCT/US93/05029) to Branham in view of appellant's admitted prior art (AAPA) and further in view of US Patent 5,649,095 to Cozza.

Group #1: Claims 5 and 15

Regarding the present group, the Examiner relies on the following excerpt from Cozza to make a prior art showing of appellant's claimed "establishing a cache buffer memory for storing upon opening of a file only a virus vulnerable portion of that file that a virus must use to enter and infect said file; said modification determining step including the steps of: indicating an open file is unmodified in the absence of an associated modification flag; responding to the presence of a modification flag by comparing a portion of said open file to the associated unmodified virus vulnerable portion of said file in said cache buffer memory to determine if the portion of the open file has been modified since the opening of the file; indicating the opened file is unmodified if the virus vulnerable portion is unmodified; and indicating the opened file is modified if the virus vulnerable portion is modified."

> "Next, in step 62, the file's cache information is checked to see if it is marked as having been previously infected by some virus which changes a file's resource fork size. If it has, then it is checked in step 64 to see if there is any difference between this file's current resource fork size and the resource fork size stored in the file's cache information. If these sizes are not equal, then flags are set in step 68 for all viruses that might cause this file's resource fork to change size when infecting. If a file's cache information is not marked as having been previously infected by some virus which changes a file's resource fork size, then the file's current resource fork size is compared with the resource fork size stored in the file's cache information in step 66 to see if they are within some predetermined tolerance. The tolerance in this step is determined based upon the size of viruses infecting a file's resource fork on the Apple Macintosh computer, upon the type of file being infected, and upon the typical size changes that might occur in Macintosh applications and other executable files due to minor changes by which the file might modify itself. This tolerance may vary from one file to another depending on file type and other factors. If these sizes are not within the predetermined tolerance, then flags are set for all viruses that might cause this file's resource fork to change size when infecting it in step 68." (col. 6, lines 21-45)

After careful review of such excerpt, however, there is clearly no suggestion of "establishing a cache buffer memory for storing upon opening of a file only a virus vulnerable portion of that file that a virus must use to enter and infect said file; said modification determining step including the steps of: indicating an open file is unmodified in the absence of an associated modification flag; responding to the presence of a modification flag by comparing a portion of said open file to the associated unmodified virus vulnerable portion of said file in said cache buffer memory to determine if the portion of the open file has been modified since the opening of the file; indicating the opened file is unmodified if the virus vulnerable portion is unmodified; and indicating the opened file is modified if the virus vulnerable portion is modified" (emphasis added). The fork information in Cozza simply does not meet the criteria of being the virus vulnerable portion of that file that a virus must use to enter and infect said file, as claimed.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has simply not been met, since the prior art references, when combined, fail to teach or suggest all the claim limitations.

In view of the remarks set forth hereinabove, all of the independent claims are deemed allowable, along with any claims depending therefrom.

VIII APPENDIX OF CLAIMS (37 C.F.R. § 41.37(c)(1)(viii))

The text of the claims involved in the appeal (along with associated status information) is set forth below:

1. (Previously Amended) A method for optimizing the operation of an anti-virus computer program for use with an operating system, comprising the steps of:

detecting a request for closure of an opened computer file;

determining in response to and after a closure request, but before file closure, if the opened computer file has been modified since being opened;

scanning said opened file for viruses before closure only if said opened file has been modified; and

closing said file if unmodified, and closing said file after scanning for viruses if found virus free.

2. (Original) The method of Claim 1, further including before said detecting step, the steps of:

determining whether said operating system includes a "dirty cache buffer" to raise or set a modification flag relative to a file being modified during the time it has been open, a computer code being indicative of said flag; and

using the computer code for a raised or set modification flag, if available, for carrying out said modification determining step by checking for the presence of a raised modification for said file.

3. (Original) The method of Claim 2, wherein if it is determined that said operating system does not provide a file modification flag, said method further includes the steps of:

establishing a "dirty cache buffer"; and

raising a modification flag in said "dirty cache buffer" if an opened file associated with said flag has been modified by a write operation.

4. (Original) The method of Claim 1, wherein said operating system includes a "dirty cache buffer" for providing a computer code for a modification flag indicative of the modification of an open file, said method further including in said modification determining step, the step of:

detecting the presence of said modification flag to determine if the associated opened file has been modified.

5. (Previously Amended) The method of claim 4, further including the steps of:

scanning a file for viruses in response to a request for opening the file;

opening said file if virus free;

establishing a cache buffer memory for storing upon opening of a file only a virus vulnerable portion of that file that a virus must use to enter and infect said file;

said modification determining step including the steps of:

indicating an open file is unmodified in the absence of an associated modification flag;

responding to the presence of a modification flag by comparing a portion of said open file to the associated unmodified virus vulnerable portion of said file in said cache buffer memory to determine if the portion of the open file has been modified since the opening of the file;

indicating the opened filc is unmodified if the virus vulnerable portion is unmodified; and

indicating the opened filc is modified if the virus vulnerable portion is modified.

6. (Original) The method of Claim 1, wherein said step of determining in response to a closing request if the opened computer file has been modified since being opened includes the step of:

monitoring network protocols to determining if a write packet was initiated for a given open file.

7. (Previously Amended) A method for optimizing operation of an anti-virus program in an operating system, said operating system including programming for raising a flag indicative of modification of an open file during the time the file has been open, said method including the steps of:

detecting the event of a request for closing said file being made to said operating system;

after said detecting, but before file closure, determining whether said modification flag has been raised by said operating system for said open file;

scanning said open file, in response to said modification flag, for viruses before permitting said operating system to close said file; and

skipping said step of scanning for viruses before closure of said open file, whenever said modification flag is not present.

8. (Previously Amended) A method for optimizing the operation of an anti-virus program in use in an operating system, said operating system including programming for raising a flag indicative of modification of an open file during the time the file has been open, said method including the steps of:

scanning a file for viruses in response to a request from an associated computer user to open and gain access to said file;

permitting said file to be opened if virus free;

storing upon opening a virus vulnerable unmodified portion of said open file;

detecting the event of a request for closing said open file being made to said operating system;

after said detecting, but before file closure, determining whether said modification flag has been raised by said operating system for said open file;

scanning said open file, in response to said modification flag, for viruses before permitting said operating system to close said file;

skipping said step of scanning for viruses before closure of said open file, whenever said modification flag is not present;

responding to the presence of a modification flag by comparing the stored unmodified virus vulnerable portion of said file to the associated portion of said open file to determine if that portion has been modified during the time the file has been open; and

skipping said step of scanning for viruses before closure of said open file if the virus vulnerable portion of said open file is unmodified.

9. (Previously Amended) A computer program product embodied on a computer readable medium for detecting computer viruses on a file server, the file server providing file storage and retrieval services for at least one client computer over a network, said computer program product comprising:

computer code for detecting an open request from a client computer, the open request asking for a requested file from the file server;

computer code for scanning said requested file for computer viruses, whereby the file server is permitted to provide said requested file to the client computer if no computer viruses are found therein;

computer code for detecting a close request from the client computer associated with said requested file;

computer code for, after said detecting, but before file closure, accessing operating system flag that indicates whether the requested file was changed prior to said close request;

computer code for scanning said requested file for computer viruses if said requested file was changed prior to said close request; and

computer code for skipping scanning said requested file if it was not changed prior to said close request.

10. (Original) The computer program product of claim 9, wherein said operating system flag is generated externally to said computer program product by the operating system in order to reduce redundant disk writes, whereby said computer code for scanning is invoked upon closing of the requested file only when actual disk writes are made by the operating system for the requested file.

11. (Original) The computer program product of claim 10, wherein said computer code for accessing uses a file handle generated by the operating system to identify the operating system flag corresponding to the requested file, said handle having been generated when the file was opened.

12. (Previously Amended) A computer program product embodied on a computer readable medium for detecting computer viruses on a file server, the file server providing file storage and retrieval services for at least one client computer over a network, said computer program product comprising:

computer code for detecting an open request from a client computer, the open request asking for a requested file from the file server;

computer code for scanning said requested file for computer viruses, whereby the file server is permitted to provide said requested file to the client computer if no computer viruses are found therein;

computer code for detecting a close request from the client computer associated with said requested file;

computer code for, after said detecting, but before file closure, accessing an operating system flag that indicates whether the requested file was changed prior to said close request;

computer code for skipping scanning said requested file if it was not changed prior to said close request;

computer code responsive to said requested file having been changed prior to said close request for determining whether a virus vulnerable portion of said file was changed;

computer code for skipping scanning said requested file if a virus vulnerable portion of said file was not changed prior to said close request; and

computer code for scanning said requested file if a virus vulnerable portion of said file was changed prior to said close request.

13. (Original) The computer program product of claim 12, wherein said

operating system flag is generated externally to said computer program product by the operating system in order to reduce redundant disk writes, whereby said computer code for scanning is invoked upon closing of the requested file only when actual disk writes are made by the operating system for the requested file.

14. (Original) The computer program product of claim 13, wherein said computer code for accessing uses a file handle generated by the operating system to identify the operating system flag corresponding to the requested file, said handle having been generated when the file was opened.

15. (Previously Presented) A method for optimizing the operation of an anti-virus computer program for use with an operating system, comprising the steps of:

detecting a request for closure of an opened computer file;

determining in response to and after a closure request, but before file closure, if the opened computer file has been modified since being opened;

scanning said opened file for viruses before closure only if said opened file has been modified; and

closing said file if unmodified, and closing said file after scanning for viruses if found virus free;

wherein said operating system includes a "dirty cache buffer" for providing a computer code for a modification flag indicative of the modification of an open file, said method further including in said modification determining step, the step of detecting the presence of said modification flag to determine if the associated opened file has been modified;

wherein further included are the steps of:

scanning a file for viruses in response to a request for opening the file,

opening said file if virus free, and

establishing a cache buffer memory for storing upon opening of a file only a virus vulnerable portion of that file that a virus must use to enter and infect said file;

wherein said modification determining step includes the steps of:

indicating an open file is unmodified in the absence of an associated modification flag,

responding to the presence of a modification flag by comparing a portion of said open file to the associated unmodified virus vulnerable portion of said file in said cache buffer memory to determine if the portion of the open file has been modified since the opening of the file,

indicating the opened file is unmodified if the virus vulnerable portion is unmodified, and

indicating the opened file is modified if the virus vulnerable portion is modified.

IX APPENDIX LISTING ANY EVIDENCE RELIED ON BY THE APPELLANT IN THE APPEAL (37 C.F.R. § 41.37(c)(1)(ix))

There is no such evidence.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 971-2573. For payment of any additional fees due in connection with the filing of this paper, the Commissioner is authorized to charge such fees to Deposit Account No. 50-1351 (Order No. NAI1P194).

Respectfully submitted,

By: ________________ Date: 9/29/04

Kevin J. Zilka
Reg. No. 41,429

Zilka-Kotab, P.C.
P.O. Box 721120
San Jose, California 95172-1120
Telephone: (408) 971-2573
Facsimile: (408) 971-4660